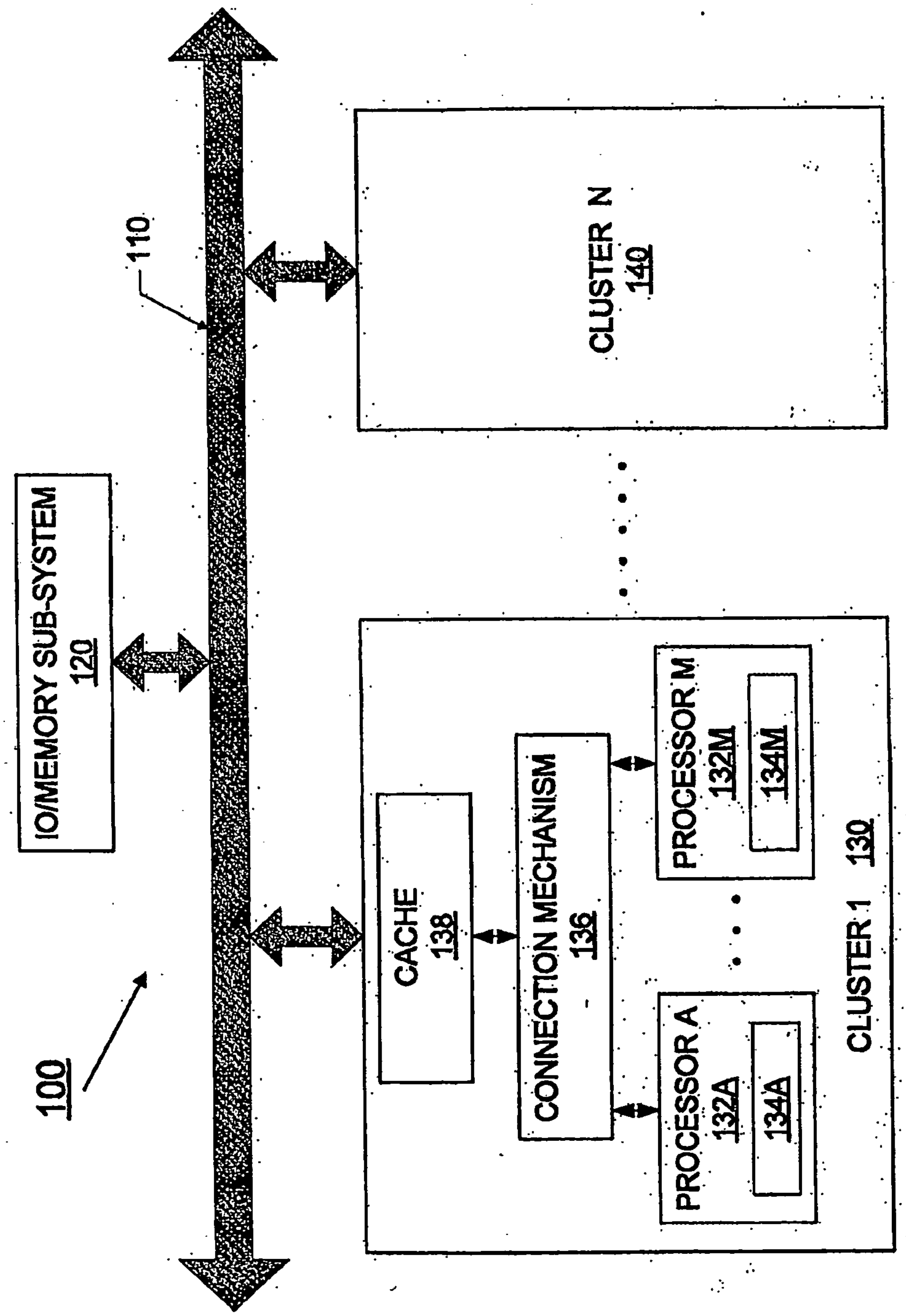
100
110
IO/MEMORY SUB-SYSTEM
120
CLUSTER N
140
CACHE
138
CONNECTION MECHANISM
136
PROCESSOR A
132A
134A
PROCESSOR M
132M
134M
CLUSTER 1
130

FIGURE 1

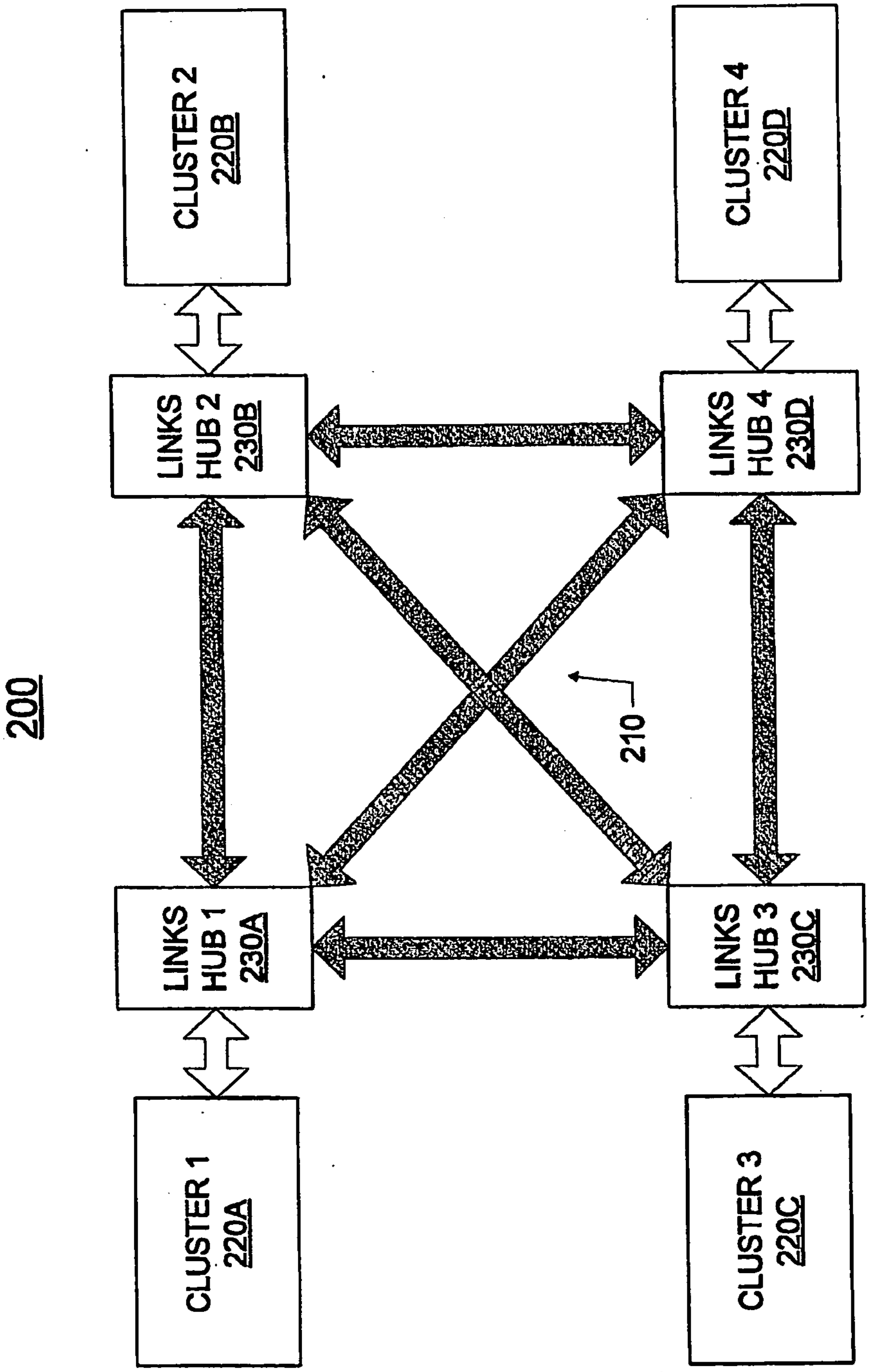
200
CLUSTER 1
220A
CLUSTER 2
220B
CLUSTER 3
220C
CLUSTER 4
220D
LINKS HUB 1
230A
LINKS HUB 2
230B
LINKS HUB 3
230C
LINKS HUB 4
230D
210

FIGURE 2

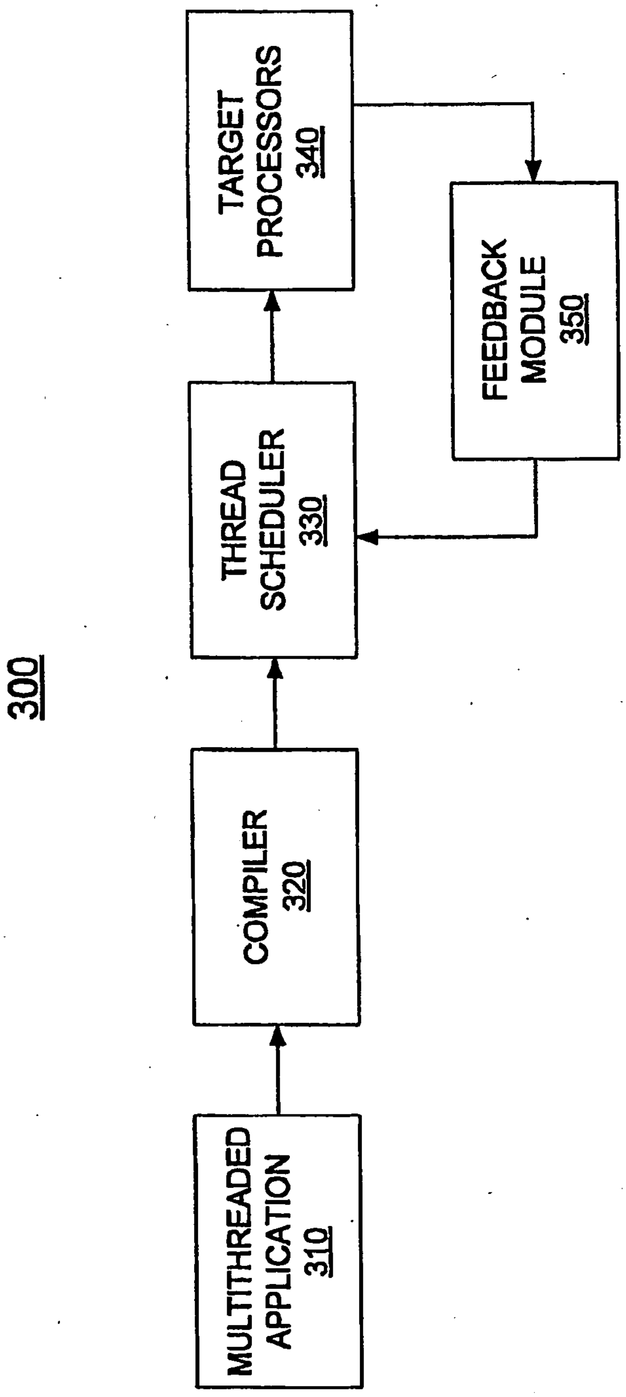
300
MULTITHREADED APPLICATION 310
COMPILER 320
THREAD SCHEDULER 330
TARGET PROCESSORS 340
FEEDBACK MODULE 350

FIGURE 3

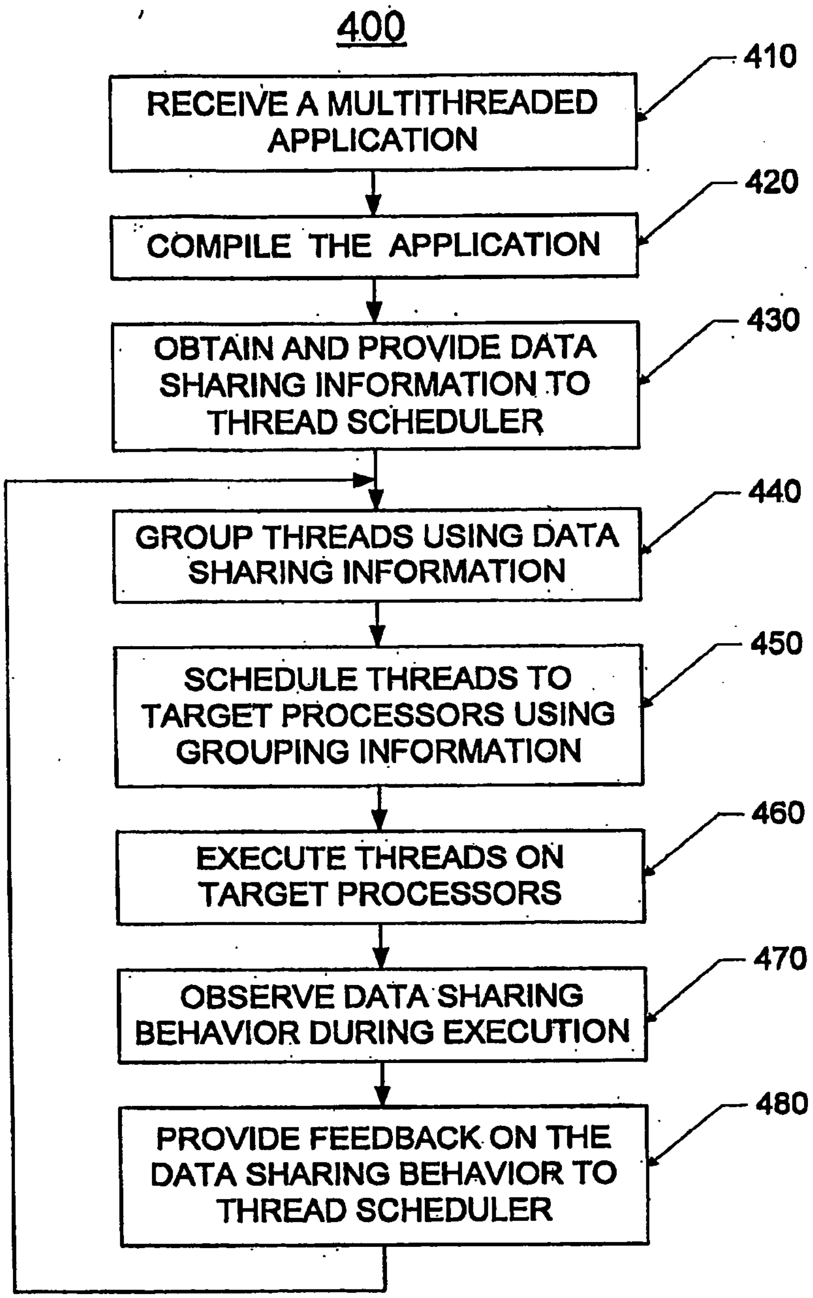
400
RECEIVE A MULTITHREADED APPLICATION
410
COMPILE THE APPLICATION
420
OBTAIN AND PROVIDE DATA SHARING INFORMATION TO THREAD SCHEDULER
430
GROUP THREADS USING DATA SHARING INFORMATION
440
SCHEDULE THREADS TO TARGET PROCESSORS USING GROUPING INFORMATION
450
EXECUTE THREADS ON TARGET PROCESSORS
460
OBSERVE DATA SHARING BEHAVIOR DURING EXECUTION
470
PROVIDE FEEDBACK ON THE DATA SHARING BEHAVIOR TO THREAD SCHEDULER
480

FIGURE 4

500

```
510   void test(List p)
520   {
530      while (p != NULL)
540      {
550         do_work(p);
560         p = p->next;
570      }
580   }
```

FIGURE 5

600

```
605   void test(List p)
610   {
615     #pragma intel omp parallel taskq shared(p) num_threads(4)
620     {
625        while (p != NULL)
630        {
635           #pragma intel omp task captureprivate(p)
640           {
645              do_work(p);
650           }
655           p = p->next;
660        }
665     }
670   }
```

FIGURE 6

700

```
705  __kmpc_serialized_parallel;
710  __kmpc_fork_call     ; create a team of threads
715  __kmpc_taskq         ;select one thread to be master thread
720  __kmpc_end_taskq     ; turn others to worker threads
725  __kmpc_end_serialzed_parallel

750  __kmpc_task_buffer         ; allocate memory for thunk_t
755  copy captureprivate variable to thunk_t
760  __kmpc_task                ; enqueuing of a task
765  __kmpc_taskq_task          ; enqueuing the task_thunk
770  __kmpc_end_taskq           ; turn into worker thread
```

FIGURE 7

800

```
805  __kmpc_serialized_parallel;
810  __kmpc_fork_call     ; create a team of threads
815  __kmpc_team_scheduling ; schedule team of threads on nearby processors
820  __kmpc_taskq         ;select one thread to be master thread
825  __kmpc_end_taskq     ; turn others to worker threads
830  __kmpc_end_serialzed_parallel

850  __kmpc_task_buffer         ; allocate memory for thunk_t
855  copy captureprivate variable to thunk_t
880  __kmpc_task                ; enqueuing of a task
865  __kmpc_taskq_task          ; enqueuing the task_thunk
870  __kmpc_end_taskq           ; turn into worker thread
```

FIGURE 8

900

```
905   int NumThreads; // specify the number of threads
910   int NumThreadsPerGroup; //specify the number of threads per group
915   int GroupNum = NumThreads / NumThreadsPerGroup;
920   #pragma omp parallel for num_threads(GroupNum)
925   {
930       for (int gid = 0; gid < GroupNum; gid++)
935       {
940           #pragma intel omp parallel taskq num_threads(NumThreadsPerGroup) ordered
              shared(frame)
945           {
950               #pragma intel omp ordered
955               frame = decoding_video();
960               #pragma intel omp task captureprivate(frame)
965               extract_feature(frame);
970           }
975       }
980   }
```

FIGURE 9

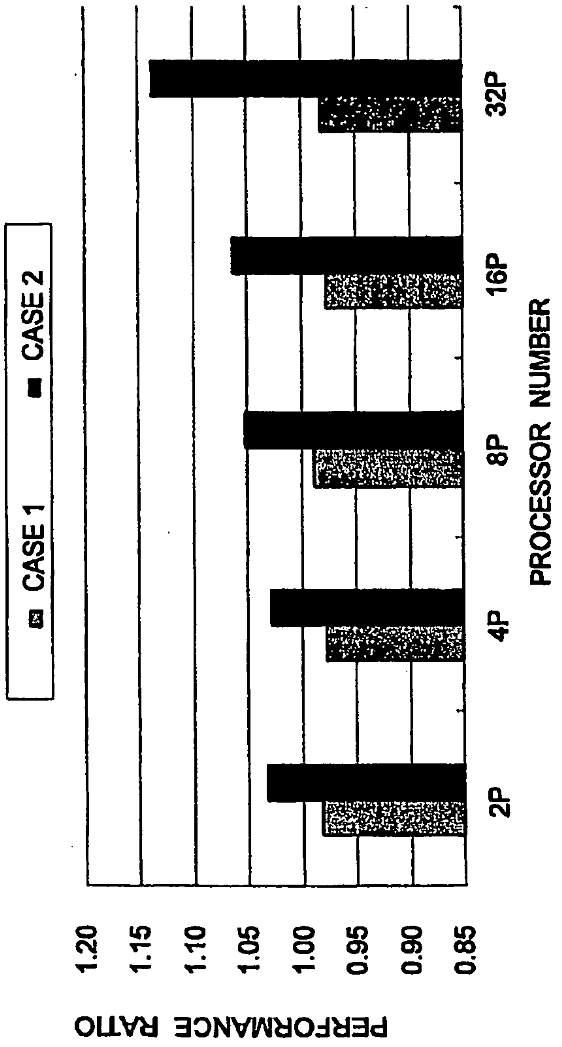
▨ CASE 1
■ CASE 2
PERFORMANCE RATIO
1.20
1.15
1.10
1.05
1.00
0.95
0.90
0.85
2P
4P
8P
16P
32P
PROCESSOR NUMBER

FIGURE 10